Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №6

Технології розроблення програмного забезпечення

Тема: Патерни проектування

Виконав
студент групи ІА-34:
Жительний В.В.

Перевірив:
Мягкий М.Ю.

Київ 2025

Мета: Вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» та навчитися застосовувати їх в реалізації програмної системи.

Тема:

21. **Online radio station** (iterator, adapter, factory method, facade, visitor, client-server)

Додаток повинен служити сервером для радіостанції з можливістю мовлення на радіостанцію (64, 92, 128, 196, 224 kb/s) в потоковому режимі; вести облік підключених користувачів і статистику відвідувань і прослуховувань; налаштувати папки з піснями і можливість вести списки програвання або playlists (не відтворювати всі пісні).

Хід роботи

Для реалізації системи "Онлайн-радіостанція" було обрано патерн проєктування "Фабричний метод". Даний шаблон доцільно використовувати в ситуаціях, коли процес створення об'єктів, у моєму випадку – медіафайлів та плейлистів є складним, містить умовну логіку або може змінюватися в майбутньому. Його застосування дозволяє відокремити код створення об'єктів від коду, що їх використовує, знижуючи зв'язність компонентів системи.

У розробленій системі цей шаблон реалізовано двома класами:

1. **AudioFileFactory**: Інкапсулює складну логіку вибору фізичного аудіофайлу на сервері. Клієнтський код запитує файл для конкретного треку з певним бітрейтом, а фабрика самостійно визначає коректний шлях до файлу, реалізуючи алгоритм пошуку та стратегію "fallback", повернення оригінального файлу, якщо потрібна версія відсутня.
2. **PlaylistFactory**: Централізує процес ініціалізації нових плейлистів. Це дозволяє задавати єдині налаштування за замовчуванням для всіх нових списків відтворення в одному місці, спрощуючи підтримку та розширення функціонала.

Графічне представлення структури реалізації даного шаблону наведено на діаграмі класів

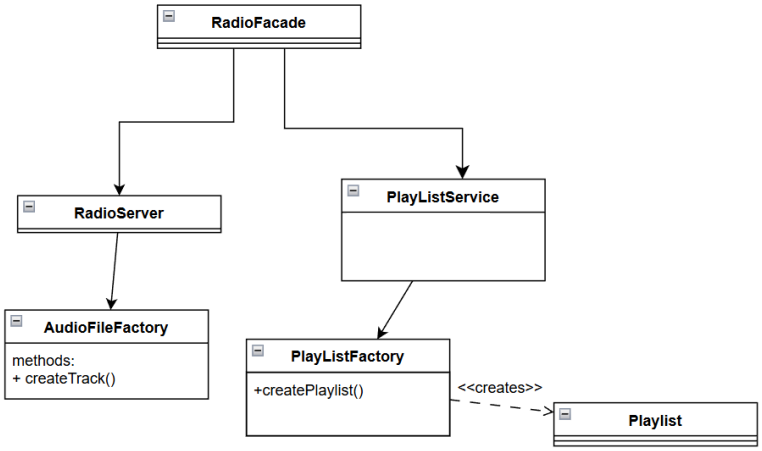


Рис. 1 Діаграма класів для шаблону Фабрика

Коротко описуючи діаграму: Центральним елементом є клас RadioFacade, який виступає точкою входу для запитів. Він взаємодіє з PlaylistService, який, у свою чергу, делегує створення об'єктів Playlist класу PlaylistFactory. Паралельно з цим модуль RadioServer ініціалізує компонент StreamProcessor, який використовує AudioFileFactory для отримання коректних файлів треків під час трансляції.

Програмна реалізація

```
@Component
public class AudioFileFactory {
    public File getAudioFile(Track track, int bitrate) {
        String basePath = track.getFilePath();

        String newPath = basePath.replace(".mp3", "_" + bitrate + "k.mp3");

        File bitrateFile = new File(newPath);

        if (bitrateFile.exists() && bitrateFile.canRead()) {
            return bitrateFile;
        }

        System.out.println(">>> Factory: Bitrate file not found: " + basePath);
        File defaultFile = new File(basePath);
        if (defaultFile.exists() && defaultFile.canRead()) {
            return defaultFile;
        }

        System.err.println(">>> Factory: No files found for track: " +
basePath);
        return null;
    }
}
```

Клас AudioFileFactory відповідає за реалізацію логіки вибору медіа-ресурсу. Метод getAudioFile приймає метадані треку та цільовий бітрейт станції. Алгоритм динамічно формує шлях до файлу, намагаючись знайти транскодовану версію (наприклад, із суфіксом _128k.mp3). У цьому методі

реалізовано стратегію що якщо спеціалізований файл для обраного бітрейту фізично відсутній на диску, фабрика автоматично повертає оригінальний файл (basePath). Це гарантує безперервність ефіру навіть у випадку, коли контент не був попередньо оброблений.

```
@Component
public class PlaylistFactory {

    public Playlist createPlaylist(String name) {
        Playlist playlist = new Playlist();
        playlist.setName(name);
        return playlist;
    }
}
```

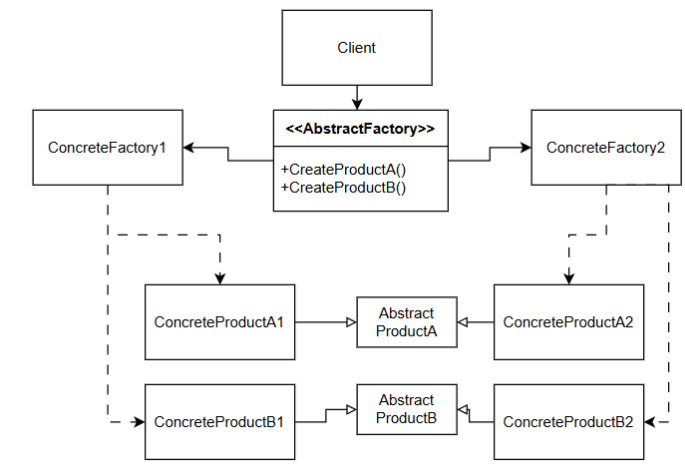
Клас PlaylistFactory реалізує породжувальний патерн для сутності плейлиста. Його основна мета – інкапсулювати процес ініціалізації нових списків відтворення. Використання фабрики дозволяє централізувати налаштування за замовчуванням для всіх створюваних плейлистів та ізолювати клієнтський код від деталей конструювання об'єктів, що спрощує подальше розширення системи, наприклад, додавання автоматичної генерації дати створення або прив'язки до власника.

Контрольні питання:

1. Яке призначення шаблону «Абстрактна фабрика»?

Призначенням шаблону є створення сімейств об'єктів без вказівки їх конкретних класів. Він структурує знання про схожі об'єкти і створює можливість повної взаємозаміни різних сімейств, наприклад перемикання роботи з SQL Server на Oracle.

2. Нарисуйте структуру шаблону «Абстрактна фабрика».



3. Які класи входять в шаблон «Абстрактна фабрика», та яка між ними

взаємодія?

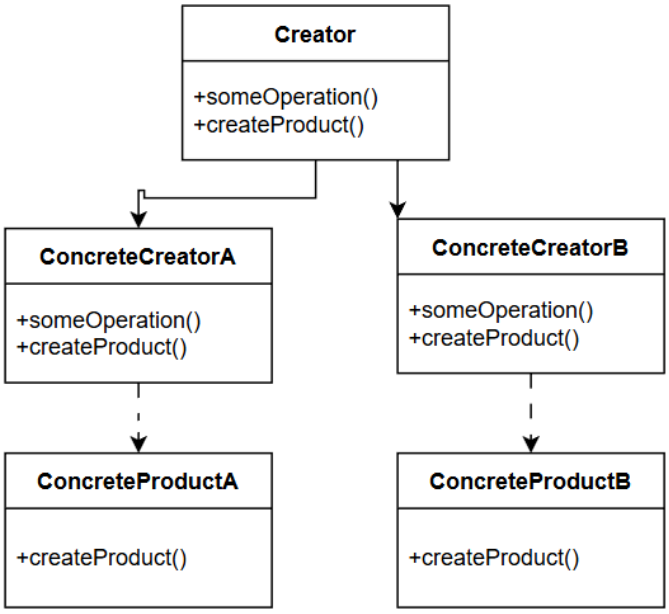
- AbstractFactory: Загальний інтерфейс фабрики, що оголошує методи створення продуктів.
- ConcreteFactory: Реалізує інтерфейс фабрики, створює об'єкти конкретного стилю.
- AbstractProduct: Інтерфейс для типів продуктів (стіл, вікно).
- ConcreteProduct: Реалізація продукту конкретного сімейства.

Взаємодія: Клієнтський код працює лише через інтерфейси. При ініціалізації клієнту передається конкретна фабрика. Коли клієнт запитує створення об'єкта, фабрика повертає продукт, що відповідає її сімейству.

4. Яке призначення шаблону «Фабричний метод»?

Шаблон визначає інтерфейс для створення об'єктів певного базового типу, але дозволяє підкласам вирішувати, об'єкт якого саме типу створювати. Це використовується, коли потрібно додати можливість створення об'єктів не базового типу, а дочірнього, зберігаючи загальну логіку роботи.

5. Нарисуйте структуру шаблону «Фабричний метод».



6. Які класи входять в шаблон «Фабричний метод», та яка між ними взаємодія?

- Creator: Базовий клас, що містить основну логіку роботи з об'єктом і оголошує фабричний метод.

- ConcreteCreator: Спадкується від Creator і реалізує фабричний метод, створюючи конкретний продукт.
- Product: Базовий клас для об'єктів.
- ConcreteProduct: Конкретні реалізації продукту.

Взаємодія: Клієнт взаємодіє з системою через базовий інтерфейс Creator. Конкретний творець у своєму фабричному методі створює та повертає конкретний продукт, підставляючи його в загальний процес обробки.

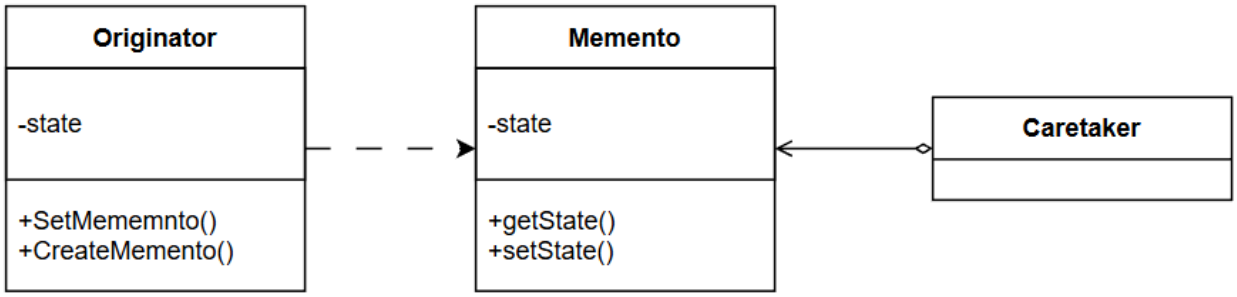
7. Чим відрізняється шаблон «Абстрактна фабрика» від «Фабричний метод»?

Головна відмінність полягає в масштабі та механізмі реалізації: «Абстрактна фабрика» призначена для створення цілих сімейств взаємопов'язаних об'єктів, наприклад усіх елементів кімнати в одному стилі, і базується на композиції, що робить її незручною для розширення новими методами, тоді як «Фабричний метод» фокусується на створенні одного продукту або його підтипів через механізм наслідування, дозволяючи дочірнім класам змінювати тип створюваного об'єкта без зміни базового коду.

8. Яке призначення шаблону «Знімок»?

Призначення – збереження і відновлення стану об'єктів без порушення інкапсуляції.

9. Нарисуйте структуру шаблону «Знімок».



10. Які класи входять в шаблон «Знімок», та яка між ними взаємодія?

- Originator: Об'єкт, стан якого треба зберегти. Тільки він має доступ до вмісту Memento.
- Memento: Зберігає стан Originator. Для всіх інших об'єктів він є закритим.

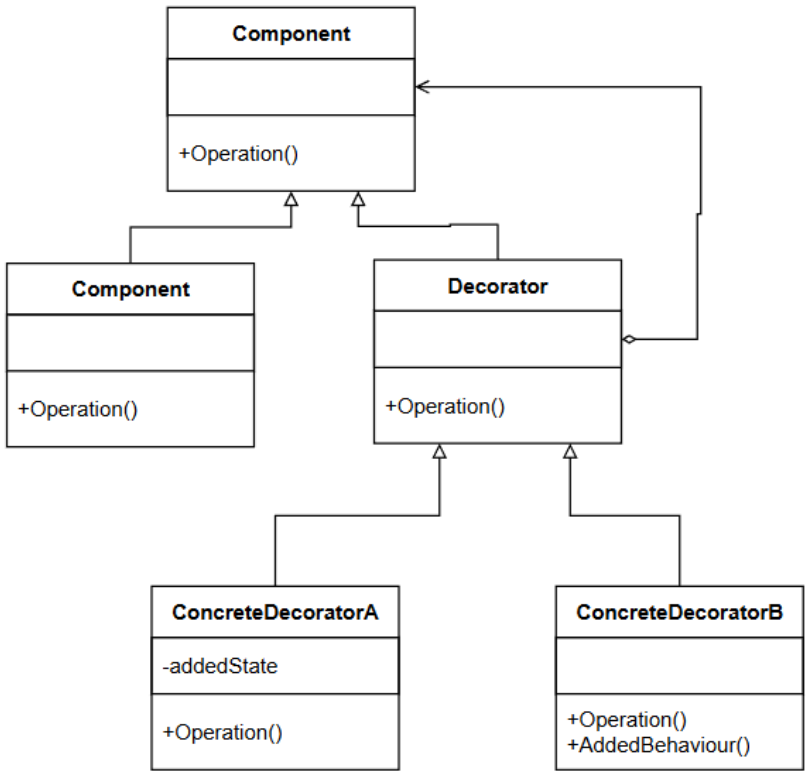
- Caretaker: Відповідає за зберігання та передачу об'єктів Memento, але не має доступу до їхнього вмісту.

Взаємодія: Originator створює Memento зі своїм станом і передає його Caretaker. Коли потрібно відновити стан, Caretaker повертає Memento назад Originator, який витягує з нього дані. Це дозволяє не зберігати історію версій всередині самого об'єкта.

11. Яке призначення шаблону «Декоратор»?

Призначений для динамічного додавання функціональних можливостей об'єкту під час роботи програми. Він дозволяє змінювати поведінку окремих об'єктів, а не всієї системи, і є гнучкішою альтернативою наслідуванню.

12. Нарисуйте структуру шаблону «Декоратор».



13. Які класи входять в шаблон «Декоратор», та яка між ними взаємодія?

- Component: Визначає інтерфейс об'єктів, наприклад, візуальний елемент.
- Decorator: Обертає Component. Має той самий інтерфейс, що й Component.
- ConcreteDecorator: Додає нову поведінку, наприклад смугу прокрутки, до або після виклику методу обгорнутого об'єкта.

Взаємодія: Декоратор містить посилання на об'єкт Component. Коли клієнт викликає метод Декоратора, той виконує свою додаткову дію і переадресовує виклик вкладеному об'єкту, або навпаки.

14. Які є обмеження використання шаблону «декоратор»?

Поява великої кількості крихітних класів та важко конфігурувати об’єкти, які загорнуто в декілька декораторів одночасно.

Висновок: Під час виконання даної лабораторної роботи було досягнуто поставлену мету: вивчено та проаналізовано групу шаблонів проєктування, що використовуються для вирішення типових задач в об'єктно-орієнтованому програмуванні. Було детально розглянуто призначення, структуру та принципи взаємодії класів для наступних патернів: «Абстрактна фабрика», «Фабричний метод», «Знімок», «Спостерігач» та «Декоратор». А також на практиці в програмному коді було реалізовано шаблон «Фабричний метод».